R14

Journal of Information Technology (1999) **14**, 107–120

Information systems prototyping in practice

PAUL BEYNON-DAVIES AND DOUGLAS TUDHOPE
School of Computing, University of Glamorgan, Pontypridd, Mid-Glamorgan CF37 1DL, UK
HUGH MACKAY
Faculty of Social Sciences, The Open University, UK

This paper describes a study of the use of prototyping approaches in information systems (IS) development. It reports a comprehensive literature review of prototyping from which we developed a taxonomy of prototyping practice. This analysis guided an empirical study which has collected data on IS practitioners' formulations of the current experience of prototyping in UK IS development organizations. These formulations are compared with data collected in a number of case studies of prototyping projects.

Introduction

In his seminal article on the foundations of software engineering, Brooks (1987) makes the analogy between software projects and werewolves. The software project is capable of becoming a monster of missed schedules, blown budgets and flawed products. However, unlike the werewolf, there is no silver bullet to make software costs drop as rapidly as hardware costs. This is because, Brooks maintains, most of the claimed solutions to the software problem attack the accidental but not the essential nature of software. By the accidental nature of software he means the processes of representation and by the essential nature of software he means the processes embodied in conceptualisation. Brooks states:

> I believe the hard part of building software to be the specification, design and testing of this conceptual construct, not the labour of representing it and testing the fidelity of the representation.

Giddings (1984) makes a similar point in characterizing information systems development as being the process of producing what he calls domain-dependent software. Information systems are domain-dependent because there is a necessary interdependence between the software system and its domain or universe of discourse. Such systems are characterized by an intrinsic uncertainty about the universe of discourse. In particular, the use of the software may change the very nature of the universe of discourse.

In this paper we survey the literature on a topic which Brooks sees not as a silver bullet but as a promising innovation which can start to address the essence of the software problem. Also, Giddings has referred to prototyping in the context of managing the uncertainty inherent in information systems development. Paul (1993) makes a similar point in asserting that most IS development works under the assumption of what he calls the fixed-point theorem. This theorem states that:

> There exists some point in time when everyone involved in the system knows what they want and agrees with everyone else.

Paul maintains that this is inherently a false assumption and questions much of systems development practice in terms of this. He particularly calls for a more evolutionary approach to development work; the idea that a system is never finished, it is a living entity which needs to continually evolve.

Interestingly, references to the use of prototyping in IS development are in some cases over twenty years old. Prototyping is clearly not a new development activity, but it is an activity which has achieved a certain prominence in recent times, particularly with the rise of rapid application development (RAD) tools and methods (Martin, 1992). Prototyping, however, is a term which is ambiguous and carries a variety of meanings.

Our aims in this paper are three-fold:

(1) To describe the reported practice of prototyping. Similar to Paul (1993) and Harel (1980) we find it useful to describe the current knowledge concerning prototyping as a folk theory: a collection of accepted wisdom. According to Harel, each of these elements of wisdom or theorems has three major characteristics: popularity, anonymous authorship, and apparent age.
(2) As is the nature of a folk theory, there is surprisingly little evidence to support any of the basic assumptions associated with prototyping practice. In this paper we report on the findings of an interview study of prototyping practice which attempted to verify empirically elements of this theory.

(3) To compare and contrast the formulations provided by practitioners of prototyping in the interview study with case studies of the practical use and adaptation of prototyping in industrial and commercial settings.

The results reported here are from a three-year study which was originally concerned with mapping the diversity of prototyping in information systems development and the specification of good practice, particularly in relation to developer–user relations. However, in the early stages of our project (Beynon-Davies *et al.*, 1996) we found that the issue of prototyping was almost invariably associated in UK developers' perceptions with the issue of RAD.

The study had three stages:

(1) A literature and documentary survey. To collate the accepted folk theory (Harel, 1980) on prototyping.
(2) To conduct a number of semistructured interviews with people in the IS development industry. This was mainly in order to answer questions of applicability and typing. Our intention was to provide a contrast of theory and practice.
(3) A number of in-depth studies of prototyping in practice were conducted. This stage involved the research group in conducting ethnographic studies of the progress of prototyping projects at organization sites from initiation to completion. Users and technical staff involved in the projects were observed both in their natural work settings and in development sessions and were regularly interviewed by the researchers.

This paper primarily reports on the results from stages 1 and 2 of our research. A review of stage 3 of our work is also included.

The structure of the paper is as follows. In the next section we attempt some preliminary definition of the terms *prototype* and *prototyping*. In the third section we provide a taxonomy of prototyping practice based on an analysis of the IS development literature on prototyping. The fourth section collates the dominant held benefits of the prototyping approach, followed by a section which discusses some of the key problems with prototyping, and highlights the lack of empirical study of this practice in natural settings. The sixth section describes the results from our interview study of prototyping and the seventh section summarizes our case study work in this area. The final section concludes with a description of some of the key issues arising from our continuing research in this area.*

*An early version of this paper was presented at the European Conference on Information Systems, Lisbon, 1996

Prototyping and prototypes

Prototyping has been discussed since the late-1970s in the IS development literature and has frequently been cast as an alternative information systems development method to the classic waterfall model (Boehm, 1976). In this context the waterfall model appears to be used as a 'straw man' to critique in terms of the benefits of prototyping. Graham (1989), for instance, cites prototyping approaches as instances of what he calls non-monolithic life-cycle development models. However, Floyd (1984) questions the positioning of prototyping as an alternative method. She maintains that 'prototyping is not, in itself, a method for systems development... It should rather be considered as one procedure within systems development that needs to be combined with others.'

It is useful to highlight the difference between the concepts of an information systems prototype and the process of information systems prototyping. Vonk (1990) defines a prototype as being, 'a working model of (parts of) an information system, which emphasises specific aspects of that system'. In contrast, prototyping is an approach to building information systems which uses prototypes. This activity frequently, but not always (see the next section), takes the following form (Dearnley and Mayhew, 1983): the information systems developer, after some initial investigation, constructs a working model which s/he demonstrates to the user. The developer and the user then discuss the prototype, agreeing on enhancements and amendments. This cycle of inspection–discussion–amendment is repeated several times until the user is satisfied with the system.

Note that, the use of the word prototype tends to suggest the tentative nature of this artefact. It is early, unfinished or a model of something. Alavi (1984) defines a prototype as being 'an early version of a system that exhibits the essential features of the later operational system'. Floyd (1984) discusses the problem that a prototype literally means 'first of a type'. This makes sense in a mass-manufacturing situation with well-defined requirements. It makes less sense in software development where normally only one product will be produced and where the desired characteristics of the product may not be well-known in advance. There is also no clear consensus about how the final prototype should relate to the final product. As we shall see in the next section, there is some debate about whether prototypes should be discarded once built. For these reasons, some people find the term prototype something of a misnomer. They propose alternative terms such as *early-version* or *simulation* to emphasize the clear use to which this system is put.

Prototyping tends to be associated in the information systems development literature with high-level tools for systems development. For instance, Budde *et al.* (1992) summarize the use of fourth generation languages (4GLs) and database-oriented tools for prototyping. More recently Internet and Intranet tools have been discussed in the context of prototyping approaches (Yourdon, 1996). However, prototyping is a technique, not just a tool (Hartson and Smith, 1991). The technique has been considered effective even when using pencil and paper or cardboard mock-up prototypes. Muller (1991), for instance, directly advocates low-technology prototyping (in an approach known as PICTIVE) as a reaction to conventional rapid prototyping environments in which 'developers have a disproportionate design impact because of the implicit politics and skill embodied in the technology'. The aim of techniques like PICTIVE is to 'empower' users by using technology they can understand and manipulate.

A taxonomy of prototyping practice

Prototyping as a method or technique within information systems development can be categorized in a number of ways. For our purposes, we find it useful to characterize this activity in terms of three questions: *what*, *when*, and *how* to prototype.

What to Prototype

Floyd (1984) distinguishes between horizontal and vertical prototyping. A horizontal prototype involves shallow (incomplete) development of all or most system functions. A vertical prototype involves deep (complete) implementation of selected features of a system. In practice the degree and extent of horizontal and vertical prototyping will reflect the trajectory of the development. For instance, it is likely that an incremental prototype will be developed in a series of horizontal and vertical moves dependent on the contingencies of a given project. Figure 1 illustrates these options in terms of four layers of a conventional information technology system.

It is important to recognize that although most of the literature addresses software prototypes, particularly user-interface prototypes and performance/capacity prototypes, business prototypes can also be constructed with very little software contribution to effectively simulate some business processes or functions (Warren,

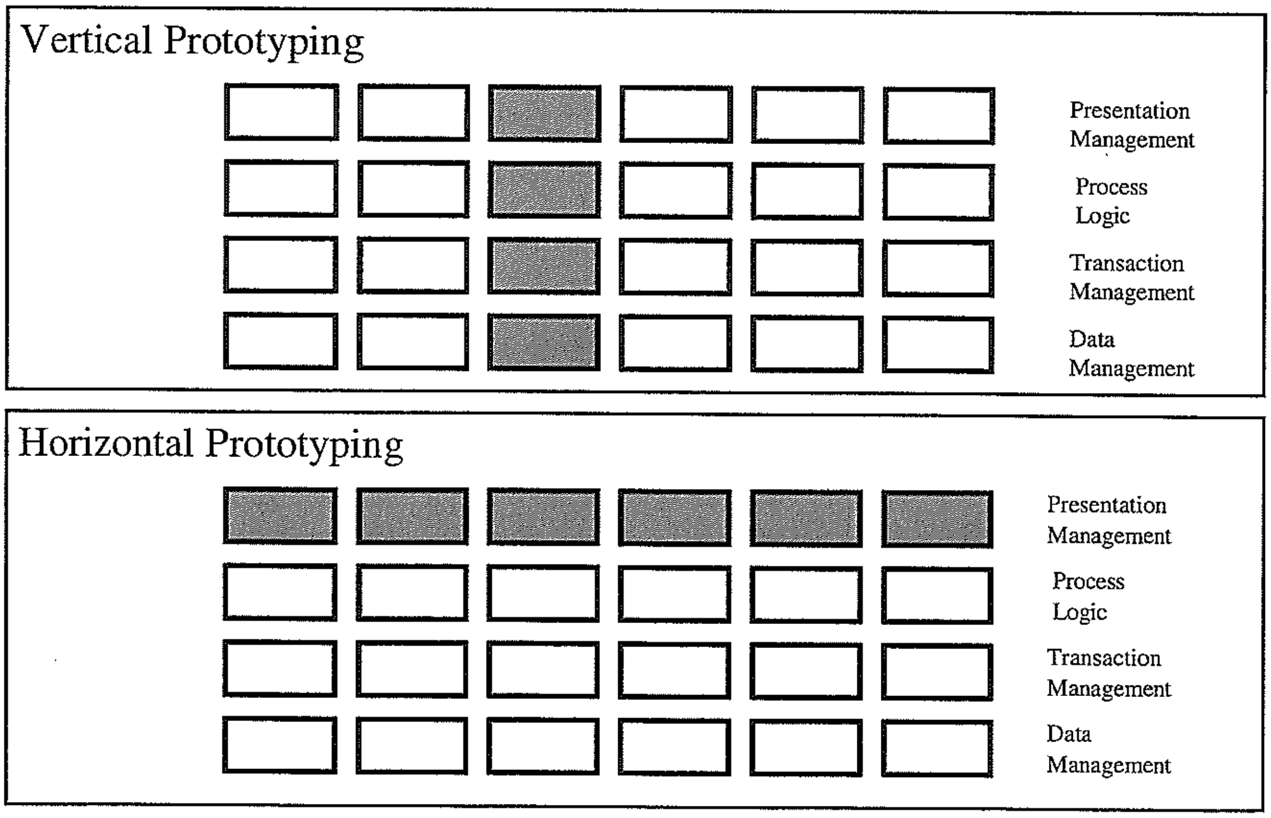

Figure 1 Horizontal and vertical prototyping

1995). There is some emphasis of this style of prototyping in the recent Business Process Reengineering (BPR) literature (Hammer and Champy, 1993). There is also much emphasis in the participatory design literature on so-called low-technology prototypes–mock-ups of information technology using resources such as flip-charts and post-it notes (Muller, 1992).

When to prototype

Whilst recognizing that some of the literature portrays prototyping as at the opposite scale to waterfall approaches, one way to distinguish between different forms of prototyping activity is on the basis of when this activity appears in the standard systems life-cycle (Ratcliffe, 1988):

(1) Early prototyping occurs at the feasibility or requirements analysis phases. Lichter *et al.* (1994) call a prototype used to initiate a software project a 'presentation prototype'. Generally, the purpose of early prototypes is to elicit or validate requirements. Vonk (1990) particularly adheres to this model of prototyping.

(2) Middle prototyping occurs at the design stage (Ratcliffe, 1988). This form of prototype is generally used to confirm the behaviour of the system in key areas, or to validate key aspects of the design.

(3) Late prototyping occurs at the implementation or even maintenance phases. This form of prototype is used to investigate key operational parameters, particularly in the area of performance. Luqi (1989) discusses the use of prototyping for evolving systems in this context.

Clearly, the use of prototyping is contingent on the nature of a specific project. Carey and Currey (1989) quote three main reasons for considering when to prototype: when the development is on-line; for all new development; whenever user expectations or requirements are unclear. Klingler (1988) maintains it is appropriate to prototype for systems that 'are usually dynamic and oriented to transaction processing and user dialogues'. The recent RAD literature has also established the type of systems appropriate for prototyping work (Stapleton, 1997). Most RAD projects are conducted on applications that are not complex computationally, highly interactive and that have a clearly defined user group.

Hardgrave (1995) (also reported in Doke *et al.*, (1992) conducted a survey amongst 118 IS managers in the United States. He identifies the top 12 variables reported as influencing the decision to use a prototyping approach:

(1) unclear requirements
(2) large systems
(3) complex systems
(4) availability of tools
(5) on-line systems
(6) project duration
(7) feasibility/testing
(8) new development
(9) user lacks DP experience
(10) need user involvement
(11) critical system
(12) developer's experience

These variables are similar to those reported in other studies. This leads Hardgrave to conclude that in practice a set of factors is being used in industry to decide when to use prototyping.

How to prototype

Three main forms of prototype are evident in the literature:

(1) A throwaway prototype is used to test out some part of a system but then discarded. Budde *et al.* (1992) call this an exploratory prototype. Prototypes for user interface design are frequently discussed in this context. For instance, Lewis *et al.* (1989) discuss the use of graphical user interface (GUI) building tools in the construction of user interface prototypes. A variant of this type is the experimental prototype (Floyd, 1984) where the aim is to determine the adequacy of some proposed system solution.

(2) An incremental prototype is one which, by incremental refinement, will form the whole of or part of a delivered system (Budde *et al.*, 1992).

(3) Graham (1989) makes the useful distinction between incremental development and incremental delivery. An incremental prototype comprises a complete system that is developed incrementally; an evolutionary prototype forms part of a proposed system which is planned to be delivered incrementally. Hence, all phases of development are conducted for the increment delivered including coding and testing before moving on to the next increment (Lichter *et al.*, 1994).

Benefits of prototyping

It has taken over twenty years for prototyping to become accepted as an IS development approach. Hardgrave (1995) cites his own and Langle *et al.*'s (1984) longitudinal data as evidence of the growth of prototyping in the IS development industry over several years. In 1984, 33% of respondents from a sample of US companies reported using prototyping. This rises to 46% usage in 1987, 49% in 1988 and 61% in 1990. In 1995, Hardgrave himself reports 71% of his respondents from a similar sample using prototyping. In a similar vein, Carey and Currey (1989) report a US survey of 250 companies across sectors in which 75% of respondents reported using prototyping techniques. Carey and Currey's work also maintains that prototyping occurs in all organizational structures, regardless of company size and type. In Gordon and Bieman's (1995) analysis of 39 case studies of prototyping, 33 of the 39 cases reported the prototyping project as a success. Twenty-two of the studies used incremental prototyping; 8 used throwaway.

Therefore, given the apparent ubiquity of this practice, it is not surprising that many benefits are claimed for using a prototyping approach:

(1) *Improvements in user communication.* Users and developers can use a prototype as a common reference point for comprehending and developing requirements (Alvai, 1984; Warren, 1995). Users are seen as being better disposed towards a system using a prototyping approach. Carey and Currey (1989) report this as being the most clearly articulated reason for using prototypes.

(2) *Better requirements.* Improved functional requirements, improved interaction requirements and easier evolution of requirements are frequent claims (Carey and Mason, 1983; Budde *et al.*, 1992; Gordon and Bieman, 1995). Prototyping, it has been claimed, can be used to flush out more than a third of redundant requirements (Warren, 1995).

(3) *Greater commitment from end-users in a project.* Alavi (1984) discusses the way in which prototyping is seen as a means of improving relationships between IS developers and users. Users apparently 'buy-in' to a prototyping project (Warren, 1995). The issue of users' 'ownership' of a system is frequently discussed in this context.

(4) *Ease of use.* Systems produced by groups using prototypes were judged easier to learn and use than those produced by other methods (Boehm *et al.*, 1984).

(5) *Better code.* In a classic experiment conducted by Boehm, code produced by prototyping projects was found to be only about 40% as large as that produced by more conventional, structured methods. This decrease in size was apparently due to the fact that prototyping developers refrained from 'copper-plating' their software (Boehm *et al.*, 1984) – devoted less time to over-engineering their technical solution. Also, perhaps surprisingly, Gordon and Bieman (1995) report that prototyping code seems to be more maintainable than code produced by other approaches.

(6) *Improvements in the development process.* Carey and Currey (1989) report that the average length of prototyping projects is short – on average about 4.25 months. This accords with recent RAD literature which mentions an average project duration of two to six months (Stapleton, 1997). Boehm's frequently cited experimental study of prototyping reports that prototyping groups accomplished their tasks with 45% less effort than groups using other methods (Boemn *et al.*, 1984). Gordon and Bieman (1995) report that in prototyping case studies analysed, a high percentage of the projects claim a decrease in development effort and an increase in user participation.

Problems with prototyping

To us as researchers it is interesting that, with prototyping, despite its ubiquity, as Nielsen and Relsted (1994) rightly state:

> There is a lack of convincing empirical evidence concerning the relationship between user participation [in development] and systems success.

Many of the published reports of prototyping are based on investigations conducted in an academic context, (e.g., Alavi, 1984), and hence have restricted validity (Lichter *et al.*, 1994). A number of survey studies of prototyping have been conducted in the US (Langle *et al.*, 1984; Carey and Currey, 1989; Hardgrave, 1995). However, even there, little or no on-site study of prototyping in natural settings appears to have been undertaken. Most of the material that is available consists of short, anecdotal experiences of prototyping projects. For instance, Carey (1990) and Lichter *et al.*, (1994) and Lichter *et al.*, (1994) are representative of this work. Klingler (1988) sums up with the statement: 'rapid prototypes and their benefits are primarily folklore'.

The case material which is available suggests the following range of problems associated with prototyping:

(1) *Different, nontraditional development tools and skills may be required.* The rapidity of prototyping and high user involvement can be stressful for

some developers (Warren, 1995) particularly because user requirements may change too rapidly. Because of differences between prototyping and conventional development, management support is seen as particularly important for prototyping projects, as well as the development of a suitable IS infrastructure for this new form of development (Budde *et al.*, 1992).

(2) *The cost of development effort*, particularly in the requirements analysis phase, may be greater. Warren (1995) quotes a consultant as saying that between 5%–10% of costs is typically added to the system costs in having early, throwaway, prototyping workshops. Dearnley and Mayhew (1984) confirm that a slightly greater cost is experienced in prototyping projects.

(3) *Project management becomes more difficult.* Prototypes can be difficult to manage and control (Alavi, 1984). Budgeting can be difficult (Gordon and Bieman, 1995). Carey and Currey (1989) speak of the struggles against creeping scope and proliferation of the number and duration of prototyping iterations. Prototypes can get bigger and functionally richer because of user enthusiasm (Warren, 1995). A related problem is that prototyping projects seldom produce detailed written specifications (Budde *et al.*, 1992). Documentation is played down in the RAD literature.

(4) *Prototypes can be oversold.* Prototyping can display the tendency to generate unrealistic requirements (Gordon and Bieman, 1995). Managers, on seeing a prototype, may be tempted to release it prematurely to market. Mount (1994) refers to the problem of managing user expectations in the context of rapid application development. This is echoed by Carey and Currey (1989). Carey (1990) cites the problem of convincing users that what they saw in an early prototype is the same as what gets finally delivered.

(5) *Difficult to prototype large systems.* Prototyping as an activity tends to be restricted to small-scale developments (Alavi, 1984; Gordon and Bieman, 1995). Few organizations in the UK appear to be building large systems using prototyping (Warren, 1995). In terms of RAD, the tendency is to rule out the applicability of this approach for large-scale, infrastructure projects such as corporate-wide databases. Indeed, some people propose that RAD projects can only be effectively produced once such infrastructure systems have been put in place.

(6) *Difficulty of recruiting users and maintaining user enthusiasm.* Prototyping depends on the willingness and ability of users to provide useful information (Alavi, 1984). User and developer enthusiasm may diminish after a working prototype is provided (Hartson and Smith, 1991). Carmel *et al.* (1993) discuss the difficulties of recruiting the right mix of user types for joint application design (JAD) sessions. They report on how users at JAD sessions tend to be limited to middle managers and not operational workers, who know the work best (Beynon-Davies *et al.*, 1997).

(7) *Difficulty of envisionment.* Recent Scandinavian material has cast doubt on the folk theorem that participation is always a good thing to have in IS development projects (Nielsen and Relsted, 1994). There is some evidence that design teams with heavy user involvement have a tendency to automate existing work practices rather than attempting to envision new ways of working. Lichter *et al.* (1994) comment, 'Many developers expect far too much from the users concerning creativity and innovative ideas about the technological design of a prototype'.

Research into prototyping practice

To summarize our analysis of the literature on prototyping, it is a development activity which is discussed in the literature with reference to two major objectives:

1. Improving the productivity of the IS development process. Most of this branch of the literature tends to address the issue of appropriate tools. It is particularly concerned with providing better support for the IS production process. 4GLs, Computer-Aided Software Engineering (CASE) and more latterly Internet and object technology are all very much part of this debate. The early literature (e.g., Lipp, 1986) on prototyping in particular takes this technological focus.
2. Improving the business effectiveness of information systems. This discussion seems to centre around the idea of improving the fit between business needs and technical objectives. One key emphasis here is the involvement of business people, particularly middle managers, in the development process. Both of these elements are key features of rapid application development (Stapleton, 1997).

As an initial generalization, we would suggest that the focus in the IS development literature seems to be moving from considerations of the productivity of development and how this may be enhanced (a technological focus), to a more open focus on the business effectiveness of information systems (a business focus). For instance, the recent emphasis on RAD (Bates and Stephens, 1995), within the practitioner communities of the UK and elsewhere, places active and regular user participation with prototype evaluation at the forefront.

Analysis of interview results

Between April and September 1995 we conducted a number of semistructured interviews with IS developers who claimed to be using prototyping at 40 UK organizations. Organizations were included from a number of sectors of commerce and industry: nine in finance, three in media, six in the public administration sector, two in the military, two in utilities, three in retail, three in transport, six in manufacturing, one in construction, and five specifically in the software industry. Our aim was not to survey the extent of prototyping in the UK in the vein of Hardgrave (1995) and Langle *et al.* (1984). Instead, it was to achieve a deeper understanding of the current experience of prototyping amongst a range of organizations within the UK: an interpretative approach much in the vein of Walsham (1993). We particularly wanted to empirically verify elements of the folk theory pertaining to prototyping discussed in previous sections, specifically the levels to which elements of the accepted wisdom concerning prototyping is reflected in practitioners' formulations. Each interviewee was therefore encouraged to talk about their experience in a relatively free-form way, although a check-list of questions and points was used to ensure coverage and comparability across interviews. A discussion of our interview results is presented below (a tabulated version of the analysis is presented in the Appendix):

Definition of prototyping

There seemed to be a lack of a clear, agreed, definition of what prototyping constitutes amongst our respondents. The most frequent formulation appeared to be one of sitting users alongside developers, building mock-up screens and using rapid development tools. This was summed up by one developer as being, 'a quick way of demonstrating fitness for purpose'. Another interviewee described it as 'from flash to bang' development. Some interviewees spoke of prototyping in terms of employing 'commonsense' and taking a pragmatic approach to IS development particularly in terms of gaining commitment from end-users. One interviewee put it this way: 'I tend to look at it [prototyping] . . . as illustrating to a customer . . . a potential system that they can get a little bit of hands-on feel to it without picking up too much in the way of a committal to the system'. A few interviewees also spoke of prototyping (particularly RAD approaches) as being nothing new, but something they were doing on mainframes twenty years ago.

Type of prototype produced

The production of screens with limited background functionality was undoubtedly the most popular form of prototype produced (by 67% of respondents); graphical user interfaces were frequently mentioned in this context.

Tools employed

Not surprisingly, most respondents cited the use of rapid application development tools such as 4 gls and GUI builders in association with DBMS as their preferred prototyping environment (71%), although one interviewee surprisingly did cite COBOL as an appropriate tool for prototyping!

Type of prototyping employed

A definite preference for incremental prototyping was evident amongst our set of respondents (58%). Throwaway prototyping normally seemed to be only considered if the users were unhappy with a system. Evolutionary prototyping was mentioned only by 3% of respondents. Interestingly, however, there is some evidence of a tension between the idea of technical quality in the developers minds and business needs in many of our interviews. The following quotations are representative:

> We would all like to throw our prototypes away and start again, because by the time you've come to the end of a system development you think that I could do that a lot better if I started again. Although we rarely get time to do that . . . users always want the system developed yesterday and to a cost . . .

> I think that the only time we would throw it away would be if it just didn't work; if it was just a can of worms . . .

How much prototyping was used

Our set of respondents was clearly a biased sample in that we sought individuals in organizations who were using prototyping. However, a large degree of variation was evident in the amount of prototyping conducted by those we contacted. Some respondents reported their organization using prototyping for all development projects (32%); others used it on as little as 5% of development projects (10%). However, in most of the organizations it was reported that prototyping approaches were formally accepted on the part of IS management (67%). The accepted approach in the organizations of those we interviewed was frequently an in-house (25%) or off-the-shelf (22%) methodology, usually adhering to some variant of a waterfall model. In such organizations a particular group was seen as championing prototyping within more conventional approaches to development.

When was prototyping used

Since most of the developers we spoke to used an incremental form of prototyping, prototyping was seen mainly as both a requirements capture technique (50%) and a design technique (20%). When indicated, prototyping seemed to be most often used on relatively small-scale (frequently PC-based rather than mainframe-based) projects, particularly new developments where the requirements were frequently vague (12%).

The benefits of prototyping

The main benefits cited by our set of developers centred around four main areas:

Better relationship with users (referred to by 52% of respondents). Under this heading we can place issues such as greater communication between developers and users, greater ownership and acceptance by user groups, better management of user expectations (45% of our interviewees referred to these issues), and risk reduction (7%). Some representative quotations are given below:

> There have been some beautifully written systems that failed because the user did not believe in them, while there have been some absolute monstrosities that have found favour because the users bought into them and believed.

> One of the things that you do find in prototyping, by the fact that you do involve users in the development process, there's more likelihood . . . of there being greater acceptance and ownership of that system, we've found.

> They [involved users] get so possessive of the system that I shudder to think of any manager who would criticise it. . . . they virtually sold it, they showed it to their manager as their solution basically, and they took a lot of pride in it which made our job a lot easier.

Better requirements (referred to by 52% of respondents). Many interviewees commented on the ability of the prototyping approach to generate better requirements. The ability to illustrate to users the 'look and feel' of a system and receive more effective feedback was felt to be a very important feature of prototyping approaches.

Better Products (referred to by 15% of respondents). Some interviewees stated that prototyping produced a better product (10%) in the sense that it better fitted with the business and was more tailorable (5%). Subjects felt that prototyping contributed to systems that better met users needs particularly in areas where the requirements were uncertain. One interviewee commented, 'The product at the end is much more appropriate to the user's needs'.

Faster and more flexible development to shorter time scales (referred to by 35% of respondents). A number of interviewees claimed that prototyping projects were faster than conventional development approaches in their experience. Others commented on the flexibility of the approach in allowing mistakes to be caught earlier. The following quotation is particularly useful:

> Traditional programming techniques assume that you do first class analysis. . . . Prototyping says that we know we're going to get it wrong at the start. . . . Prototyping to me is recognizing that people are not perfect. While traditional systems design makes the assumption that people are perfect and that people will give you the correct answers and, of course, they don't

Many of the benefits of prototyping commented on by our interviewees are usefully summarized in the following quotation:

> If you go off to buy . . . I don't know . . . a car, or something like that . . . you can say, well OK, this is the model I want, I want this colour and all the rest of it . . . And then you place the order in expectation of receiving that thing . . . If however you receive a different model, a different colour and all the rest of it you'd be rather disappointed . . . It's rather similar in building a system . . . If you have described to the people who are going to receive the system what the scope of the system is to do, and later you describe it in more detail, and they are involved in that process by virtue of prototyping . . . then not only have your expectations been managed because you know what you're going to get . . . you have also seen it in advance and therefore will probably have a greater sense of ownership . . .

Problems with prototyping

The main problems discussed in our set of interviews can be located in three areas:

Difficulty of managing the user. The problem of excessive expectations was frequently mentioned (30%). The potential for the system to be oversold or users buying-in too early relates to this issue. As one interviewee put it: 'The largest single problem in prototyping is that it raises expectations . . . expectation levels get very high . . . keeping those up as the system goes along is a problem.'

A number of interviewees referred to the potential for proliferation of user requirements (22%). 'You get an educated audience out there now, the users, they see things on the TV and they say that we want a part of that . . .' One developer referred to this as the 'butterfly mentality' of users meaning they flitted from one requirement to another. Some mention was also made of the difficulty of envisionment on the part of users.

User Reluctance. User reluctance to participate fully was formulated frequently as a problem (32%). One interviewee made the comment that if users do not turn up to development meetings, 'that's really not good enough, it's their system and they should have ownership of it'. Another interviewee acknowledged the difficulties of getting users involved: 'The main problem is that you're asking the business to commit some of its best people to the development process for up to four months.' Frequently, organizations seemed to offer user involvement but did not formally commit user time to the project. Given this situation it is not surprising to find user reluctance: 'When you're suddenly asked to join in all the time, and carry on your normal job as well, then that presents a bit of difficulty'.

Project management. The difficulty of fitting prototyping approaches with conventional project management was mentioned as a problem by some interviewees (17%). One interviewee commented, 'Iterative development does not sit well with conventional project management techniques'. Another said, 'The major problem is to provide a persuasive justification as to the cost of prototyping'. Difficulty of adequately scoping a prototyping project, determining when to terminate iterative development and getting formal sign-offs from users were some of the reasons cited for this difficulty:

(a) The difficulty is where do you call the end-point.
(b) The thing is with prototyping, you've got to have a much, much better handle on what your scope is . . . otherwise, it's [the system] going to run away.
(c) Prototyping is more difficult to get formal sign-offs . . . It's more difficult because you don't have lots of documentation that you can sign off against.

User involvement

User involvement was generally seen by practitioners as central to the development process (73%). 'If users aren't involved and they don't appreciate what the system is going to look like, then you might as well not bother'. Frequently this was seen as important for increasing feelings of ownership, better requirements and a better product. A general problem of getting the 'right' users was frequently mentioned (Beynon-Davies *et al.*, 1997). The term 'right' was frequently phrased in terms of people empowered to make design decisions and knowledgeable about the business processes. Interestingly, the most frequent characterization of operational or end-users (referred to by one interviewee as 'nuts and bolts' users) was as people to whom you would demonstrate the system in the later stages of a project. Interestingly, some respondents alluded to the difficulty of preventing technically-aware and enthusiastic end-users from driving a project in ways detrimental to other user groups.

Methods and project management

Only 22% of interviewees discussed using an 'off-the-shelf' method for prototyping. 25% of our respondents were using either some in-house method or melding prototyping with waterfall approaches based on in-house methodologies similar in nature to SSADM or Information Engineering. Many of our respondents seemed to take a much more formal approach to prototyping than that discussed in the literature. Minuted meetings and focused sessions in 'clean rooms', 'time-boxing', and formal sign-offs, key elements of the RAD approach, were frequently referred to. There is some evidence that this is due to the need to control projects within an already established organizational IS management framework. It may also accord with some developing awareness of RAD approaches amongst our sample (Stapleton, 1997).

Evaluation of success of projects

Most subjects seemed to think that employing prototyping contributed to the 'success' of IS projects. However, methods of evaluation seemed variable. Two organizations use formal usability laboratories for evaluation. For most of the organizations, evaluating the success of projects seemed to be conducted either through informal post-implementation meetings with users (25%) or via formal sign-offs (42%).

A summary of the results of the interview study

In summary, the most commonplace practice seems to be the use of incremental prototyping on relatively small-scale (usually PC-based) projects. Much use is being made of rapid application development environments on these platforms to develop screens which can be rapidly verified by users. Organizations appeared to be generally cautious in their use of prototyping, commonly attempting to meld it with more conventional systems development and project management approaches. Generally, our subjects reiterated many of the benefits of prototyping which are discussed in the literature, particularly better relationships with end-users, better requirements and products, and more rapid and flexible development. Some expressed scepticism concerning the applicability of some proposed methodologies for incremental development.

Interviewees also voiced some of the perceived problems of the prototyping approach, particularly difficulties of managing user expectations, user reluctance and project management problems. In most organizations business sector involvement seems to be encouraged, but end-user participation in design and decision-making in the senses advocated in the recent participatory design literature (Kuhn and Muller, 1993) is absent. This was seen by many of the developers themselves as a problem. Related to this, a commonly experienced difficulty is finding the 'right' type of user to participate in development. Many organizations seemed to take a formal approach to evaluating the 'success' of projects while other development groups seemed to rely on informal feedback from the business users.

At this stage, in a similar manner to Hardgrave (1995), it is impossible to discover how much of this constitutes a straightforward rationalization of practice on behalf of respondents in terms of the professional literature. Interestingly, however, many respondents displayed scepticism as to the applicability of IS development methodologies in general and the new iterative methodologies in particular; indicating a degree of critical awareness on the part of our set of interviewees. It is therefore not surprising to find that, in most of the organizations we have studied we found a contingent approach to information systems methods, and prototyping is no different in this respect.

Case studies

Evidence from our interview study suggests that the contemporary development community, at least in the UK, formulates the applicability of prototyping within a more encompassing rapid application development approach.

RAD first became topical in the IS development community with the publication of a text by James Martin with the same title (Martin, 1992). Martin defines the key objective of RAD as: *the commercial need to deliver working business applications in shorter timescales and for less investment.* In the UK there has recently been an attempt to promote a standard RAD approach. The Dynamic Systems Development Method consortium have now produced version 3 of a public domain RAD method (Consortium 1995).

In the third phase of our study we collected data on seven RAD projects within UK organizations. In this section we highlight the way in which prototyping was used in the projects studied. More detail of the case study data is reported in Beynon-Davies (1998).

Most of the projects studied concerned themselves with developing some form of information system, generally with low to medium backend complexity and high levels of front-end interactivity. Most were also based on some form of RDBMS tool framework, although Internet/Intranet technology was also a popular development medium.

Prototyping seems to have been an inherent feature of all projects, as was user involvement. However, whereas user involvement was usually on a periodic basis, developer involvement was normally dedicated to a specific project. There was also quite a variation of user-developer interaction styles with a mix of both formal (scheduled meetings) and informal interaction (unscheduled developer-user interaction) being commonplace.

In terms of prototyping, incremental software prototypes which served as requirements elicitation, systems design and systems implementation vehicles were evident. However, one project involved the production of a paper prototype of a process model. In-situ modification of prototypes was evident only in one of the projects studied. Changes to prototypes were most frequently made after meetings with users.

We observed some use of low-technology prototyping on one of the studied projects. However, generally there was little evidence of attempts on the part of project participants to envision new ways of working. Also, there was a conspicuous lack of awareness of the participatory design literature on the part of the developers we spoke to.

In many of our cases RAD was regarded as a new and innovative approach to IS development. Frequently one particular group in the organization was championing this approach as an alternative to an in-house, waterfall-based methodology. In at least two of the studied organizations there was some evidence that RAD in general and prototyping in particular was experiencing some success in diffusing throughout the development organization.

In general terms most of the project participants we have interviewed claim that clear benefit is derived from deploying RAD. As an indicator of this, all of the projects were completed within their allotted time and budget. However, it has proven difficult to asssess the relationship between the use of RAD and the success of the eventual product of the development process. This is mainly because, for all but two of the projects, post-implementation data is scarce*. In the two projects where such data is available, the systems are being used

*This accords with Kumar's finding (Kumar, 1990) that post-implementation evaluation of information technology systems are rarely done.

and there is evidence of considerable commitment to the system on the part of the user community.

Conclusion

Prototyping is normally characterized as a contrasting approach to waterfall methods of IS development. In recent years, the approach has enjoyed something of a renaissance, particularly in association with rising interest in RAD. We have reported some key elements of the IS development literature on prototyping. Our analysis of this material suggests that the focus of the development literature has moved over the last few years from a technological to a business-oriented focus. The early material on prototyping tends to be subsumed under the interest in higher-level tools for development; the recent material has tended to emphasise the importance of user involvement and rapid development to support business change.

This change in emphasis seems to be reflected in an analysis of our interview material in which practitioners are now frequently reiterating to us the importance of user-participation in development and casting such development in terms of 'ownership' of systems by users and 'empowerment' of users in the development process. They are also frequently reiterating the importance of development work being driven by and reacting to business need.

Further research work has been conducted by our research group on prototyping. We studied two development projects in-depth using ethnographic techniques and collected case data in relation to five other projects. These observational studies were designed to probe further the issue of how close the performance of prototyping matches the expectations expressed by practitioners. Interestingly, prototyping nowadays in the UK is commonly understood by practitioners in terms of RAD. We therefore shifted our focus slightly from an analysis of prototyping *per se* to a more encompassing analysis of the practicalities of RAD as a development approach. Our main interest has been in gaining a better understanding of the negotiated order that arises in RAD projects. Our observations have revealed a number of ways in which project groups made up of both developers and users formulate and organise their work. Great play is made of using artefacts such as 'to-do' lists as a collective form of group memory. Although not explicitly framed as such by project participants, we see many similarities here with some of the low-technology prototype emphasis in the area of participatory design. Hence, there appears to be much potential in a detailed consideration of how participatory design may contribute to the contemporary commercial emphasis on RAD.

Acknowledgements

This project was funded by the UK Economics and Social Science Research Council (Grant No: R000 23 5505). We would like to thank all our respondents who discussed their prototyping work with us.

References

Alavi, M. (1984) An assessment of the prototyping approach to information systems development, *CACM*, 27(6), 556–63.

Bates, P. and Stephens, M. (1995) Rapid application development–concept, methodology or what? *How Rapid is Rapid? New Directions in Software Development Seminar*, University of Wolverhampton, (University of Wolverhampton/BCS).

Beynon-Davies, P. (1998). Rapid application development: an empirical assessment, in Avison, D. and Edgar-Nevill, D. (eds) *UK Academy of Information Systems Conference*, University of Lincoln, (McGraw–Hill) pp. 127–40.

Beynon-Davies, P., Mackay H., Tudhope, D.S. and Slack, R. (1996) Prototyping in information systems development: a study of development practice in natural settings, *European Conference on Information Systems*, Lisbon, Portugal, pp. 953–69.

Beynon-Davies, P., Mackay, H. and Tudhope, D.S. (1997) User involvement in information systems development: the problem of finding the 'right' user, *European Conference on Information Systems*, Cork, Eire, pp. 659–75.

Boehm, B.W. (1976) Software Engineering, *IEEE Transactions on Computers*, 25(12), 24–60.

Boehm, B.W., Gray, T.E. and Seewald, T. (1984) Prototyping vs specifying: a multi-project experiment, *IEEE Transactions Software Engineering*, 10(3), 290–333.

Brooks, F.P. (1987) No silver bullet: essence and accidents in software engineering, *IEEE Computer*, 20(4), 10–19.

Budde, R., Kautz K., Kuhlenkamp, K. and Zullighoven, H. (1992) *Prototyping: An Approach to Evolutionary System Development*, (Springer-Verlag, Berlin).

Carey, J.M. (1990) Prototyping: alternative information systems development methodology, *Information and Software Technology*, 32(2), 119–26.

Carey, T.T. and Mason, R.E.A. (1983) Information systems prototyping: techniques, tools and methodologies, *INFOR – Canadian Journal of Operational Research and Information Processing*, 21(3), 177–91.

Carey, M. and Currey, J.D. (1989) The Prototyping Conundrum Datamation 1st June pp. 29–33.

Carmel, E., Whitaker R.D. and George, J.F. (1993) PD and joint application design: a transatlantic comparison, *Communications of ACM*, 36(4), 40–8.

Consortium, D. (1995) *Dynamic Systems Development Method* (version 2), Ashford, Kent.

Dearnley, P.A. and Mayhew P.J. (1983) In favour of system prototypes and their integration into the systems development cycle, *Computer Journal*, 26(1), 36–42.

Dearnley, P.A. and Mayhew P.J. (1984) On the use of software development tools in the construction of data processing prototypes, *Approaches to Prototyping*, in Budde, R., Kuhlenkamp, K. and Zulligohoven, H. (eds) (Springer-Verlag, Berlin), pp. 24–32.

Doke, E.R., Swanson, N.E. and Hardgrave, B. (1992) *The Decision to Prototype Information Systems*, Proceedings, National Decision Sciences Institute, San Francisco, CA.

Floyd, C. (1984) A systematic look at prototyping, in *Approaches to Prototyping*. Budde, R., Kuhlenkamp, K. and Zulligohoven, H. (eds) (Springer-Verlag, Berlin), pp. 12–20.

Giddings, R.V. (1984) Accommodating uncertainty in software design, *Communications ACM*, 27(5), 428–34.

Gordon, V.S. and Bieman, J.M. (1995) Rapid prototyping: lessons learned, *IEEE Software*, 30(1), 85–95.

Graham, I. (1989) Incremental development: review of non-monolithic life-cycle development models, *Information and Software Technology*, 31(1), 7–20.

Hammer, M. and Champy, J. (1993). *Reengineering the Corporation: a Manifesto for Business Revolution*, (Nicholas Brearley, London).

Hardgrave, B.C. (1995) When to prototype: decision variables in industry, *Information and Software Technology*, 37(2), 113–18.

Harel, D. (1980) On folk theorems, *Communications ACM*, 23(7), 379–89.

Hartson, H.R. and Smith E.C. (1991) Rapid prototyping in Human-Computer Interface Development, *Interacting with Computers*, 3(1), 51–91.

Klingler, D.E. (1988) The ten commandments of prototyping, *Journal of Information Systems Management*, **Summer**, 66–72.

Kuhn, S. and Muller, M.J. (1993) Participatory design, *CACM*, 36(4), 26–8.

Kumar, K. (1990) Post implementation evaluation of computer-based information systems: current practices, *CACM*, 33(2), 236–52.

Langle, G.B., Leithheiser, R.L. and Naumen, J.D. (1984) A survey of applications systems prototyping in industry, *Information and Management*, 7, 273–84.

Lewis, T.G., Handloser, F., Bose, S. and Young, S. (1989) Prototypes from standard user interface management systems, *IEEE Computer*, 22(5), 51–9.

Lichter, H., Schneider-Hufschmidt, M. and Zullighoven, H. (1994) Prototyping in industrial software projects – bridging the gap between theory and practice, *IEEE Transactions Software Engineering*, 20(11), 825–32.

Lipp, M.E., (ed.) (1986) *Prototyping: State of the Art Report*, (Pergamon, London).

Luqi (1989) Software evolution through rapid prototyping, *IEEE Computer*, 22(5), 13–25.

Martin, J. (1992) *Rapid Application Development*. (Prentice-Hall, Englewood Cliffs).

Mount, C. (1994) *Practical Aspects of Rapid Application Development*, Evolutionary Prototyping and Participatory Development, Unicom Seminars, Heathrow, London.

Muller, M.J. (1991) *PICTIVE–an Exploratory in Participatory Design*, Computers and Human Interaction, Brighton.

Muller, M.J. (1992) Retrospective on a year of participatory design using the PICTIVE technique, *CHI'92*.

Nielsen, J.F. and Relsted, N.J. (1994) A new agenda for user participation: reconsidering the old Scandinavian prescription, *Scandinavian Journal of Information Systems*, 6(2), 3–20.

Paul, R. (1993) Why users cannot get what they want, *SIGOIS Bulletin*, 14(2), 8–12.

Ratcliffe, B. (1988) *Early and Not-so-early prototyping – rationale and told support*. International Computer Software and Applications Conference (IEEE Computer Society Press).

Stapleton, J. (1997) *DSDM – Dynamic Systems Development Method: the Method in Practice*, (Addison-Wesley, Harlow, UK).

Vonk, R. (1990) *Prototyping: the Effective Use of Case Technology*. (Prentice-Hall, Englewood-Cliffs, NJ).

Walsham, G. (1993) *Interpreting Information Systems in Organisations*. (John Wiley, Chichester).

Warren, L. (1995) Reverting to Prototype Computing 25 May, pp. 34–35.

Yourdon, E. (1996) Java, the web, and software development, *IEEE Computer*, 44(2), 206–31.

Biographical notes

Paul Beynon-Davies is currently Reader in Information Systems at the University of Glamorgan. He has written a number of books on information systems topics and regularly publishes Journal papers, Conference Papers and Professional articles on issues concerning information systems development. Dr Beynon-Davies has conducted research in the areas of database systems, hypermedia systems, information systems development and information management.

Doug Tudhope is currently a Senior Lecturer in the School of Computing at University of Glamorgan, where he teaches in hypermedia, mutlimedia and HCI. His research interests include the sociology of computing, knowledge-based hypermedia and multimedia systems, thesaurus-based retrieval and information science. He is Editor of the *New Review of Hypermedia and Multimedia*.

Hugh Mackay is Senior Lecturer and Staff Tutor in Sociology at the Open University. He researches and publishes in the sociology of technology, on computer system design and new media technologies.

Address for correspondence: Paul Beynon-Davies, School of Computing, University of Glamorgan, Pontypridd, Mid-Glamorgan CF37 1DL, UK. E-mail: PBDAVIES@GLAMORGAN.AC.UK.

Appendix: Tabulation of results

Definition of prototyping (% of subjects referring to each feature)

Requirements elicitation/Triggering feedback/ Proving a concept	42%
Right systems/Fitness for purpose	25%
Building systems with user involvement/look and feel	20%
Specification of core elements	12%
Ability to get things wrong early	12%
Rapid development	12%
Referred to method	7%
Iterative development	5%

Kinds of prototype produced

Screen layouts (limited functionality)	67%
Mixture	22%
Performance	5%
Mockups (post-its)	3%
Not indicated	3%

Tools used

Rapid development (PCs/4gls/GUIs/DBMS)	71%
Not specified	17%
Conventional (Mainframes/3GLs/Files)	7%
Mixture	5%

Type of prototyping employed

Incremental	58%
Mixture	25%
Throwaway	7%
Not specified	7%
Evolutionary	3%

Prototyping regarded within organization

Formally accepted	67%
Not specified	17%
Tacitly acknowledged	10%
Informally accepted	3%
Not accepted	3%

When was prototyping used

Early (requirements capture)	50%
Middle (design)	20%
Not specified	15%
Throughout	10%
Late (production)	5%

Applicability of prototyping

Not specified	85%
Small projects	12%
Medium projects	3%

Proportion of projects using prototyping

High	32%
Project dependent	25%
Not specified	23%
Very low	10%
Low	5%
Medium	5%

Benefits of prototyping (% of subjects referring to each feature)

Better requirements	52%
Better communication and relationship with user community	45%
Rapid/Flexible development	35%
Right/Better products	10%
Risk reduction	7%
Tailorability	5%

Problems of prototyping (% of subjects referring to each feature)

Requires intensive involvement/Right users	32%
Over-inflated expectations/Over-selling	30%
Proliferation/fuzziness of requirements	22%
Project management (Terminating project/Documentation)	17%
Not specified	17%
No problems experienced	10%
Mis-communication/Difficulty of envisionment	7%

Value of User Involvement

Important	53%
Very important	20%
Not specified	15%
Not important	7%
Ambivalent	5%

When do users get involved

Early (requirements capture)	50%
Throughout (early, middle and late)	32%
Not specified	10%
Late (production)	5%
Periodically	3%

Methods used for prototyping

Not specified	40%
In-house	25%
Off-the-shelf	22%
None	10%
Mixture	3%

How are projects evaluated

Formal (Sign-offs/ Formal meetings)	42%
Informal (User feedback/Use)	25%
Not indicated	18%
Mixture	15%